Applying Key Operators in the Observable Class: Case Study ex1

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science
Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Lesson

• Case study ex1 explores calls to Observable just(), fromArray(), fromCallable(), fromIterable(), map(), doOnNext(), mergeWith(), repeat(), & blockingSubscribe() to create, reduce, multiply, & display Big Fraction objects synchronously

```java
Observable
  .just(BigFraction.valueOf(100, 3),
       BigFraction.valueOf(100, 4),
       BigFraction.valueOf(100, 2),
       BigFraction.valueOf(100, 1))
  .map(fraction -> fraction
       .multiply(sBigReducedFraction))
  .blockingSubscribe
    (fraction -> sb.append(" = "
                             + fraction.toMixedString()
                             + "\n"),
     error -> sb.append("error"),
    () -> BigFractionUtils
         .display(sb.toString()));
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Reactive/Observable/ex1
Applying Key Operators in the Observable Class: Case Study ex1
Applying Key Operators in the Observable Class to ex1

This class shows how to apply RxJava features synchronously to perform basic Observable operations, including fromCallable(), repeat(), just(), map(), mergeWith(), and blockingSubscribe().

```java
/**
 * Test BigFraction multiplication using a synchronous Observable stream.
 */

public class ObservableEx {

    public static Completable testFractionMultiplication1() {
        StringBuilder sb =
            new StringBuilder(">> Calling testFractionMultiplication1()\n").

            // Use just() to generate a stream of big fractions.
            .just(BigFraction.valueOf(numerator: 100, denominator: 3),
                 BigFraction.valueOf(numerator: 100, denominator: 4),
                 BigFraction.valueOf(numerator: 100, denominator: 2),
                 BigFraction.valueOf(numerator: 100, denominator: 1))

            // Use map() to multiply each element in the stream by a constant.
            .map(fraction -> {
```

See [github.com/douglasraignschmidt/LiveLessons/tree/master/Reactive/Observable/ex1](https://github.com/douglasraignschmidt/LiveLessons/tree/master/Reactive/Observable/ex1)
End of Applying Key Operators in the Observable Class: Case Study ex1